Homework 1 - IS 605 FUNDAMENTALS OF COMPUTATIONAL MATHEMATICS

Christophe Hunt

February 4, 2017

Contents

1 Problem Set 1

1.1 (1) Calculate the dot product $u.v$ where $u = [0.5; 0.5]$ and $v = [3; -4]$

```
u <-  c(.5, .5)
v <-  c(3 ,-4)

dotProduct <- function(x, y)
if (length(x) != length(y)){
  return("vectors are not equal lengths")
  } else {
    product <- c()
      for (i in 1:length(x)){
           product <- append(product, (x[[i]] * y[[i]]))
       }
  return(sum(product))
}

dotProduct(u,v)
```

```
## [1] -0.5
```

1.2 (2) What are the lengths of u and v?

Please note that the mathematical notion of the length of a vector is not the same as a computer science definition.

```
u <-  c(.5, .5)
v <-  c(3 ,-4)

print(paste0("the length of $u$ = ", round(sqrt(dotProduct(u,u)), 2)))
```

[1] "the length of u = 0.71"

```
print(paste0("the length of $v$ = ", round(sqrt(dotProduct(v,v)), 2)))
```

[1] "the length of v = 5"

1.3 (3) What is the linear combination: $3u - 2v$?

```
u <-  c(.5, .5)
v <-  c(3 ,-4)

linearCombo <- function(xMulti, x, yMulti, y, subtract = TRUE){
      xResults  <- c()
      yResults  <- c()
      linCombo <- c()
    for (i in 1:length(x)){
      xResults <- append(xResults, (xMulti * x[[i]]))
    }
    for (i in 1:length(y)){
      yResults <- append(yResults, (yMulti* y[[i]]))
    }
    if (subtract == TRUE){
    for (i in 1:length(xResults))
      linCombo <- append(linCombo, (xResults[[i]] - yResults[[i]]))
    } else {
    for (i in 1:length(xResults))
      linCombo <- append(linCombo, (xResults[[i]] + yResults[[i]]))
    }
    return(linCombo)
}

x <- linearCombo(xMulti = 3, u, yMulti = 2, v)

paste0("[",x[1], " , " , x[2], "]")
```

```
## [1] "[-4.5 , 9.5]"
```

1.4 (4) What is the angle between u and v

```
angle <- acos((dotProduct(u,v) / (sqrt(dotProduct(u,u)) * sqrt(dotProduct(v,v)))))
angle
```

```
## [1] 1.712693
```

2 Problem Set 2

Set up a system of equations with 3 variables and 3 constraints and solve for x. Please write a function in R that will take two variables (matrix A & constraint vector b) and solve using elimination. Your function should produce the right answer for the system of equations for any 3-variable, 3-equation system. You don't have to worry about degenerate cases and can safely assume that the function will only be tested with a system of equations that has a solution. Please note that you do have to worry about zero pivots, though. Please note that you should not use the built-in function solve to solve this system or use matrix inverses. The approach that you should employ is to construct an Upper Triangular Matrix and then back-substitute to get the solution. Alternatively, you can augment the matrix A with vector b and jointly apply the Gauss Jordan elimination procedure.

Please test it with the system below and it should produce a solution $x = [-1.55, -0.32, 0.95]$

$$\begin{bmatrix} 1 & 1 & 3 \\ 2 & -1 & 5 \\ -1 & -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 6 \end{bmatrix} \tag{1}$$

```
A = matrix(c(1, 2, -1, 1, -1, -2, 3, 5 ,4), nrow=3,  ncol=3)
b = matrix(c( 1, 2, 6), nrow = 3, ncol = 1)

sysEqSolver <- function(A, b){
      M <- cbind(A,b)

      if(all(M[,1] == 0) | all(M[,2] == 0) | all(M[,3] == 0)){
        return(print("one variable is = 0"))
      }

      while (M[1,1] == 0){
        M <- M[c(2,3,1), ]
      }

        M[2,] <- M[2,]*(M[1,1] / M[2,1])
        M[2,] <- M[1,] - M[2,]

      if(all(M[,1] == 0) | all(M[,2] == 0) | all(M[,3] == 0)){
        return(print("one variable is = 0"))
      }

      while (M[2,1] == 0 & M[2,2] == 0){
           M <- M[c(1,3,2),]
      }

      if (M[3,1] != 0){
          M[3,] <- M[3,]*(M[1,1] / M[3,1])
```

```r
        M[3,] <- M[1,] - M[3,]
      }

      if (M[3,2] != 0){
        M[3,] <- M[3,]*(M[2,2] / M[3,2])
        M[3,] <- M[2,] - M[3,]
        M[3,] <- M[3,] / M[3,3]
      }

    #backwards substitution
    M[2,4] <- M[2,4] - (M[2,3] * M[3,4])
    M[2,3] <- 0
    M[2,] <- M[2,] / M[2,2]

    M[1,4] <- M[1,4] - (M[1,3] * M[3,4])
    M[1,3] <- 0
    M[1,4] <- M[1,4] - (M[1,2] * M[2,4])
    M[1,2] <- 0
    M[1,] <- M[1,] / M[1,1]

    return(M)
  }

sysEqSolver(A,b)
```

```
##      [,1] [,2] [,3]       [,4]
## [1,]    1    0    0 -1.5454545
## [2,]    0    1    0 -0.3181818
## [3,]    0    0    1  0.9545455
```

My solution has possible improvements such as a more dynamic matrix pivoting operation, this solution is only for a 3 variable, 3 equation system. In the future, I would like to improve this function to allow for larger systems.

```r
results <- sysEqSolver(A,b)
print(paste0("The solution is $x$ = [ ", round(results[1,4], 2), ", ",
             round(results[2,4], 2), ", ",
             round(results[3,4], 2), " ]"))
```

[1] "The solution is x = [-1.55, -0.32, 0.95]"